SYSTEM AND METHOD
FOR SECURE INSTALLATION AND OPERATION OF SOFTWARE

BACKGROUND OF THE INVENTION

5   Field of the Invention

This invention pertains generally to methods for installing software and, more particularly, to methods for the secure installation and operation of software applications. The invention also relates to systems for the secure installation and operation of software.

10   Background Information

Known software installation products for Microsoft® Windows®-based platforms are believed to be designed to be generic in nature and do not take advantage of media-specific properties that are not common across all media types. Furthermore, known software products, in general, are believed to be designed to be

15   generic in nature and do not take advantage of media-specific properties that are not common across all media types.

Alternate Data Streams (ADSs) have been available since the NT File Structure (NTFS) was first introduced. In NTFS, everything on disk is a file. For example, the metadata (e.g., eleven metadata files, such as $MFT, the Master File

20   Table, are created in which NTFS stores data associated with disk management) is stored as a set of files. The Master File Table (MFT) is an index of every file on the NTFS logical volume. For each file, the MFT keeps a record containing attributes of information about the file.

It is known to employ an ADS as a secondary storage mechanism

25   when file-specific MFT data exceeds 1,500 bytes. This is an automatic process within the Windows® operating system. It is also known for this operating system to automatically employ ADSs to store thumbnails of all image files stored on an NTFS logical volume. This process is enabled by default, and can be disabled through modification of the system registry. It is believed that no installer has been written to

30   take advantage of NTFS ADSs because Windows®-based installation programs are written to work on FAT, FAT32 and NTFS volumes. Also, FAT and FAT32 volumes do not support ADSs.

Installation logs, application configuration files and other secure data may readily be accessed, archived, compressed, backed up, viewed, scanned, modified, manipulated or deleted by employing conventional techniques, such as, for example, standard system tools. Hence, these files are subject to possible accidental

5   access, storage, modification, use or deletion. Moreover, these files are readily accessible to a user who might improperly seek to intentionally access, archive, view, modify, use or delete the contents. Accordingly, there is room for improvement in methods and systems for the secure installation and operation of software.

SUMMARY OF THE INVENTION

10   These needs and others are met by the invention, which provides a method for an installer to write a Primary Data Stream (PDS) file to an NT File Structure (NTFS) logical volume, associate data with the Primary Data Stream file, and write the associated data to the NT File Structure logical volume as an Alternate Data Stream file from that installer. This allows installation logs, application

15   configuration data, and other application-specific and/or secure information to be safely hidden from view and protected from unsafe deletion or modification.

In accordance with one aspect of the invention, a method for secure installation and operation of software comprises: employing an NT File Structure logical volume; employing an installer; writing a Primary Data Stream file to the NT

20   File Structure logical volume from the installer; associating data with the Primary Data Stream file; and writing the associated data to the NT File Structure logical volume as an Alternate Data Stream file from the installer.

The method may include selecting the data from the group comprising an installation log, an application configuration file, an error log, help information,

25   and database information.

The method may include employing as the Primary Data Stream file an executable file.

The method may include creating a Primary Data Stream directory chain; writing the Primary Data Stream directory chain to the NT File Structure

30   logical volume from the installer; writing the Primary Data Stream file to the Primary Data Stream directory chain in the NT File Structure logical volume from the installer; associating the data with the Primary Data Stream directory chain or the

Primary Data Stream file by creating and closing the Alternate Data Stream file; and installing the associated data to the NT File Structure logical volume as the Alternate Data Stream file from the installer.

The method may include employing an installation file; defining in the installation file a Primary Data Stream directory chain, the Primary Data Stream file, the Alternate Data Stream file, and at least one information file; displaying the at least one information file from the installation file; creating the Primary Data Stream directory chain in the NT File Structure logical volume; copying the Primary Data Stream file from the installation file to the Primary Data Stream directory chain in the NT File Structure logical volume; and copying the Alternate Data Stream file from the installation file to the Primary Data Stream directory chain in the NT File Structure logical volume.

As another aspect of the invention, a computer system for secure installation and operation of software comprises: a processor; a first drive adapted for access by the processor; a second drive adapted for access by the processor, the second drive including an NT File Structure logical volume; and an installer operatively associated with the first drive, the installer cooperating with the processor to write a Primary Data Stream file to the NT File Structure logical volume, associate data with the Primary Data Stream file, and write the associated data to the NT File Structure logical volume as an Alternate Data Stream file.

The NT File Structure logical volume may include a directory chain or a system directory. The installer may install the Primary Data Stream file in the directory chain or the system directory of the NT File Structure logical volume.

The installer may cooperate with the processor to create a Primary Data Stream directory chain, to write the Primary Data Stream directory chain to the NT File Structure logical volume, to write the Primary Data Stream file to the Primary Data Stream directory chain in the NT File Structure logical volume, to associate the data with the Primary Data Stream directory chain or the Primary Data Stream file, and to install the associated data to the NT File Structure logical volume as the Alternate Data Stream file.

The processor may include a display. The NT File Structure logical volume may include a display utility. The display utility may cooperate with the

processor to display the associated data from the Alternate Data Stream file in the NT File Structure logical volume on the display.

The installer may include an installation file including a Primary Data Stream directory chain, the Primary Data Stream file, the Alternate Data Stream file, and at least one information file. The installer may cooperate with the processor to display the at least one information file from the installation file to the display, to create the Primary Data Stream directory chain in the NT File Structure logical volume, to copy the Primary Data Stream file from the installation file to the Primary Data Stream directory chain in the NT File Structure logical volume, and to copy the Alternate Data Stream file from the installation file to the Primary Data Stream directory chain in the NT File Structure logical volume.

BRIEF DESCRIPTION OF THE DRAWINGS

A full understanding of the invention can be gained from the following description of the preferred embodiments when read in conjunction with the accompanying drawings in which:

Figure 1 is a flowchart of steps for the secure installation and operation of software in accordance with an embodiment of the invention.

Figure 2 is a block diagram of a computer system for secure installation and operation of software in accordance with another embodiment of the invention.

Figure 3 is a block diagram of the NT File Structure logical volume and the installer of Figure 2.

Figure 4 is a flow chart of an installation process to create one or more Primary Data Stream (PDS) directory chains, to install one or more PDS files, and to install one or more Alternate Data Stream (ADS) files through association with the PDS directory chains and/or the PDS files in accordance with another embodiment of the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The invention is described in association with an installer for a DVD or compact disc (CD) drive (*e.g.*, a CD-ROM drive), although the invention is applicable to a wide range of different media and/or different drives.

Referring to Figure 1, a flowchart 1 shows the steps for the secure installation and operation of software. First, at 2, an NT File Structure (NTFS) logical volume 4 is employed. Next, at 6, an installer 8 is employed. Then, at 10, a Primary Data Stream (PDS) file 12 is written to the NTFS logical volume 4 from the installer

5   8. Next, at 14, data 16 is associated with the PDS file 12. Finally, at 20, the associated data 16 is written to the NTFS logical volume 4 as an Alternate Data Stream (ADS) file 22 from the installer 8.

Example 1

An application program 24 may be employed to write, read or

10   otherwise manipulate the ADS file 22 from such application program after writing the associated data 16 to the NTFS logical volume 4 at step 20, above.

Example 2

In addition to writing the (insecure) PDS file 12 to the NTFS logical volume 4 from the installer 8 at step 10, above, directory data (not shown) for that file

15   12 may also be written to the NTFS logical volume 4.

Example 3

Installation logs, application configuration files and other secure data may be associated with one or more PDS (e.g., executable files) by writing the secure data as the NTFS ADS file 22. Associated files written as ADSs cannot be viewed

20   using conventional techniques, cannot normally be deleted without deleting the associated visible PDS file 12, and normally can only be accessed by knowing the naming convention used to store the data 16. ADS files, such as file 22, can, however, be written to, read from and otherwise manipulated from within an originating software application, such as application program 24.

25   Example 4

It is possible to delete associated ADS file(s) from PDS files (or PDS directories) from the command line by copying the PDS file (or PDS directory chain) to a FAT or FAT32 volume (not shown), and then copying the data from the FAT or FAT32 volume back to the NTFS logical volume, which overwrites the original data.

30   Since FAT and FAT32 volumes do not support Alternate Data Streams, they will ignore any ADS file that is associated to the PDS file (or PDS directory) during the copy. Normally, Windows® installations including NTFS logical volumes rarely have

FAT or FAT32 volumes attached. Additionally, most users will not realize that ADS file(s) have been associated to PDS files (or PDS directories) and will not try to circumvent the file protection that has been implemented.

This "feature" of losing ADS file(s) associated to PDS files (or PDS
5   directories) during a copy to a FAT/FAT32 volume is of great value from a security standpoint. While it is rare to find a FAT/FAT32 volume attached to an NTFS logical volume, it is easy for a user to copy data to a FAT floppy disk (not shown), or copy data to an external hard drive or Flash RAM drive (*e.g.*, through a USB port) (not shown). Here, only PDS files can be copied to a floppy disk.

10   If an external NTFS hard drive is connected to a workstation (not shown), then the user will be able to copy over PDS files and all associated ADS files. However, transporting and connecting an external hard drive is fairly visible; hence, the risk of an unscrupulous or unknowledgeable user doing this in secret is minimal.

Only PDS files can be copied to an external USB Flash RAM drive
15   (*e.g.*, card) (not shown) because the native file system is FAT and the overhead of NTFS is too high for such a limited device.

Taken as a whole, the ability to strip ADS files from PDS files during a copy to a FAT or FAT32 volume is a valuable security feature.

<div align="center">Example 5</div>

20   It is also possible to decouple ADS file(s) from PDS files (or PDS directories) from the command line under Windows® 2000 and XP with the following three commands: (1) copy PRIMARY.EXE TEMP.EXE; (2) type TEMP.EXE >PRIMARY.EXE; and (3) del TEMP.EXE.

The first command line makes a copy of the file PRIMARY.EXE,
25   which is assumed to have associated ADS file(s). The second command line 'views' the PDS component of the file TEMP.EXE in a way that overwrites the file, PRIMARY.EXE, that has the associated ADS file(s). This will result in a new copy of PRIMARY.EXE that lacks the associated ADS file(s). Finally, the third command line simply deletes the temporary file, TEMP.EXE.

30   <div align="center">Example 6</div>

Third party applications are available that navigate through NTFS logical volumes to seek out PDS files and PDS directories that have associated ADS

files. If a user is technically savvy about NTFS logical volumes and ADS files, then they can also find the utilities that will help them locate PDS files with associated ADS files. It is believed that relatively very few users know about Alternate Data Streams under NTFS. This includes software engineers, system administrators, power

5   users and regular users. Hence, the risk of having a user view, delete or modify data stored as a regular PDS file is far greater than that of having an ADS file associated to a PDS file or PDS directory. Hence, the value of the disclosed method and computer system is considerable from a security standpoint.

Example 7

10   Descriptive information about all files and directories (not shown) within the NTFS logical volume 4 are located in a Master File Table (MFT) 26. Each MFT file/directory entry (not shown) contains a data attribute (not shown) that is used to house pointers to ADS files, such as file 22. In taking this approach of associating a PDS file, such as file 12, or directory (not shown) with zero or more ADS files, such

15   as file 22, through internal referencing through pointers (not shown), ADS files are effectively made invisible, and less prone to deletion and modification using standard system tools (not shown).

Figure 2 shows a computer system 30 for secure installation and

20   operation of software. The computer system 30 includes a suitable platform, such as a personal computer 32, workstation (not shown) or server (not shown), having a processor (μP) 34 and the Microsoft® Windows® operating system 36 (e.g., without limitation, Windows® XP Professional; Windows® XP Home; Windows® Server 2003; Windows® Small Business Server 2003; Windows® Storage Server 2003;

25   Windows® 2000 Server; Windows® 2000 Professional; Windows® NT Workstation; Windows® NT Server; future versions of Windows® that support Alternate Data Streams).

The computer system 30 also includes a first drive, such as a DVD or CD-ROM drive 38, adapted for access by the processor 34, and a second drive, such

30   as a hard disk 40, adapted for access by the processor 34. The second drive 40 includes an NTFS logical volume 42. The computer system 30 further includes an installer 44 operatively associated with the first drive 38. The installer 44 cooperates

with the processor 34 to write a PDS file 46 to the NTFS logical volume 42, to associate data 48 with the PDS file 46, and to write the associated data 48 to the NTFS logical volume 42 as an ADS file 50.

The personal computer 32 includes a display 52 that is controlled by
5   the processor 34. The PDS file 46 may be, for example, an executable file, which is adapted for execution by the processor 34.

Example 8

For example, the installer 44 of Figure 3 creates a PDS file 46′, writes data to that PDS file 46′, closes the PDS file 46′, creates an ADS file 50′ associated
10   to the PDS file 46′ (or PDS directory), writes data 48′ to the ADS file 50′, and then closes the ADS file 50′. The association process occurs when the ADS file 50′ is first created, and completes when the ADS file 50′ is closed.

Continuing to refer to Figure 3, the disk drive 40 with the NTFS
15   logical volume 42 and the CD-ROM drive 38 with the installer 44 and an associated installation file 58 are shown. The installer 44 copies a directory chain 54 (*e.g.*, without limitation, C:\Program Files\Company_Name\Electronic_Handheld_Device, C:\Program Files\Company_Name\Electronic_Handheld_Device_JDE_3.7) or system directory (*e.g.*, without limitation, C:\WINNT) from the installation file 58 to the
20   NTFS logical volume 42 as directory chain 54′ or system directory. The NTFS logical volume 42 also includes a PDS file 46′, which is copied by the installer 44 from the PDS file 46 of the drive 38. The file 46 is part of or is referenced by the installation file 58. Similarly, the NTFS logical volume 42 further includes an ADS file 50′ having data 48′, which file 50′ is copied by the installer 44 from the ADS file
25   50 of the drive 38. The file 50 is part of or is referenced by the installation file 58. In this manner, the installer 44 installs the PDS file 46′ and the ADS file 50′ in that directory chain 54′.

In summary, the installer 44 cooperates with the processor 34 of Figure 2 to create the PDS directory chain 54′, to write the PDS directory chain 54′ to the
30   NTFS logical volume 42, to write the PDS file 46′ to the PDS directory chain 54′ in the NTFS logical volume 42, to associate the data 48′ with the PDS directory chain

54´ or the PDS file 46´, and to install the associated data 48´ to the NTFS logical volume 42 as the ADS file 50´.

The NTFS logical volume 42 includes the application program 24, which cooperates with the processor 34 of Figure 2 to write, read or manipulate the
5   ADS file 50´. The NTFS logical volume 42 may also include a suitable display utility 56, which cooperates with the processor 34 to display the associated data 48´ from the ADS file 50´ in the NTFS logical volume 42 on the display 52.

As shown, the installation file 58 includes the PDS file 46, the ADS file 50, installation instructions 60, the PDS directory chain 54, an End User License
10   Agreement (EULA) 62, and one or more other information files 64.

The installer 44 cooperates with the processor 34 of Figure 2 to display the installation instructions 60, the EULA 62 and/or the one or more information files 64 from the installation file 58 to the display 52.

Figure 4 shows a flowchart 70 of an example installation process,
15   which employs the installation file 58 of Figure 3. First, at 72, one or more PDS directory chains, such as 54 of Figure 3, one or more PDS files, such as 46, one or more ADS files, such as 50, and one or more information files, such as 60,62,64, are defined in the installation file 58. Next, at 74, one or more of the information files, such as 60,62,64, are displayed from the installation file 58. For example,
20   installation-specific information, such as the EULA 62, is displayed for the user that is conducting the installation. Then, at 76, one or more PDS directory chains, such as 54, are created in (or otherwise written to) the NTFS logical volume 42 from the installer 44 as PDS directory chains, such as 54´. Then, at 80, one or more PDS files, such as 46, are installed in the one or more created directory chains, such as 54´,
25   and/or within the one or more PDS system directories. For example, the PDS files, such as 46, are written or copied as the PDS files, such as 46´, to the corresponding PDS directory chains, such as 54´, in the NTFS logical volume 42 from the installer 44. Finally, at 82, one or more ADS files, such as 50, are installed (or otherwise copied) to the NTFS logical volume 42 from the installer 44, by association with the
30   corresponding PDS directory chains, such as 54´, or the corresponding installed PDS files, such as 46´.

In summary, as shown by the flowchart 70, the installer 44 creates one or more PDS directory chains 54´, installs one or more PDS files 46´, and installs one or more ADS files 50´ through association with the PDS directory chains 54´ and/or the PDS files 46´.

5

Example 9

Although the installation file 58 of Figure 3 and the installation flowchart 70 of Figure 4 are shown, a wide range of installation files and/or installation sequences may be employed. For example, a wide range of instructions may be encoded within one or more installation file(s) in order to detail how to

10   perform the installation. For example, this may detail what file(s) to display during the process, what PDS directory or directories to create, what PDS file(s) to copy to a PDS directory or directories, and what ADS file(s) to associate with what PDS file(s) and PDS directory or directories.

Example 10

15   Many common types of files may be stored as an ADS file associated to a PDS directory chain or to a PDS file. Examples include: (1) installation logs used to alter or de-install software installations; (2) application configuration files used to set the working characteristics of a software environment; (3) error logs created within a software environment; (4) help information (*e.g.*, files; documentation) that

20   could be reverse-engineered by competitors; (5) files containing information that could be used by a competitor; (6) database information (*e.g.*, tables; views; associated indexes) that are directly accessed within the software environment; and (7) files critical to the operation of the software environment that, if written as PDS files, could be duplicated to make illegal copies of such environment.

25

Example 11

The Appendix, below, shows example software written in Microsoft® Visual C++ project form. This software file (*e.g.*, 'ads_demo.c'), when suitably compiled and executed (*e.g.*, as 'ads_demo.exe'), creates a PDS file (*e.g.*, 'ads_demo.bin'), and then creates two associated ADS files (*e.g.*,

30   'ads_demo.bin:ads_demo.log' and 'ads_demo.bin:ads_demo.cfg').

As a result, only a single file (*e.g.*, entitled 'ads_demo.bin') is visible in the working directory after running the program, 'ads_demo.exe'. Also, the file

'ads_demo.bin' has two ADSs attached (*e.g.*, 'ads_demo.bin:ads_demo.log' and 'ads_demo.bin:ads_demo.cfg').

Example 12

Although Example 11 shows a .bin file as the PDS file, a wide range of PDS files may be employed. For example, the PDS file may be an executable file, such as 'ads_demo.exe', or a directory chain.

Example 13

The contents (*e.g.*, the associated data 48′ from the ADS file 50′ in the NTFS logical volume 42 of Figure 3) of each ADS file, such as the two ADS files (*e.g.*, 'ads_demo.bin:ads_demo.log'; 'ads_demo.bin:ads_demo.cfg') of Example 11, may be viewed using the Cygwin™ CAT utility by the following three commands: (1) cat ads_demo.bin; (2) cat ads_demo.bin:ads_demo.log; and (3) cat ads_demo.bin:ads_demo.cfg. Cygwin™ tools are marketed by Red Hat, Inc. of Research Triangle Park, North Carolina.

Example 14

Alternatively, CrucialADS, which is marketed by Crucial Security of Fairfax, Virginia, may be employed to detect Alternate Data Streams in NTFS files and directories.

Example 15

Alternatively, there are several other ways to view the contents of ADS, if the exact name of the stream is known. For example, the contents can be piped to the Windows® more command (*e.g.*, C:\>more < afile.txt:hidden.txt) to provide, for example, "this is my hidden data" from the corresponding ADS file.

Example 16

As an alternative to the DVD or CD-ROM drive 38, many installations occur after downloading an installer from an external drive or source, such as the Internet. The downloaded installer may either directly install a software environment from within the downloaded file, or may install a software environment by automatically downloading components from the Internet.

Example 17

As another alternative, a common area may be provided on a local communication network to store software installers. Users of that network, in turn,

can install applications on their local system by running the installer from a network drive. No CD-ROM or DVD drive is required in these instances.

The disclosed method and system allow installation logs, application
5   configuration data and any other application-specific information to be safely hidden from view and protected from unsafe deletion or modification by an installer, such as installer 44. In switching to an ADS model for software deployment, licensed users and software manufacturers will reduce the risk of data theft, data corruption, software environment failure due to system alteration, and unlicensed duplication of
10   licensed material. The greatest potential value of ADS storage of data is to hide files to such a degree that they cannot be directly archived, compressed, backed up, viewed, scanned, altered or deleted.

While for clarity of disclosure reference has been made herein to the exemplary display 52 for displaying installation information, it will be appreciated
15   that such information may be stored, printed on hard copy, be computer modified, or be combined with other data. All such processing shall be deemed to fall within the terms "display" or "displaying" as employed herein.

While specific embodiments of the invention have been described in detail, it will be appreciated by those skilled in the art that various modifications and
20   alternatives to those details could be developed in light of the overall teachings of the disclosure. Accordingly, the particular arrangements disclosed are meant to be illustrative only and not limiting as to the scope of the invention which is to be given the full breadth of the claims appended and any and all equivalents thereof.

Appendix

```
//

// Filename: ads_demo.c

// Purpose  : demonstrate creation of primary and alternate data streams

//

5

#include <windows.h>

#include <stdio.h>


void main( void )

10   {

         HANDLE   hPDS;

         HANDLE   hADS;

         DWORD    Retval;


15       char      * Primary      = "ads_demo.bin";

         char      * ADS_log      = "ads_demo.bin:ads_demo.log";

         char      * ADS_cfg      = "ads_demo.bin:ads_demo.cfg";


         char      * Primary_out  = "ADS Demo Primary Data Stream";

20       char      * ADS_log_out = "ADS Demo Alternate Data Stream-Installation Logs";

         char      * ADS_cfg_out = "ADS Demo Alternate Data Stream-Configuration";


         //

         // create a primary data stream for demonstration purposes

25       // in real life conditions, associate an ADS file with installed executable

         //

         hPDS = CreateFile

         (

             Primary,

30           GENERIC_WRITE,

             FILE_SHARE_WRITE,
```

```
                NULL,

                OPEN_ALWAYS,

                0,

                NULL

5       );


        if   ( hPDS == INVALID_HANDLE_VALUE )
        {
                printf( "Unable to create [%s]\n", Primary );
10      }
        else
        {
                WriteFile( hPDS, Primary_out, strlen(Primary_out), &Retval, NULL );
                CloseHandle( hPDS );
15
                printf( "Primary Data Stream [%s] Has Been Created\n", Primary );


                hADS = CreateFile
                (
20                  ADS_log,
                    GENERIC_WRITE,
                    FILE_SHARE_WRITE,
                    NULL,
                    OPEN_ALWAYS,
25                  0,
                    NULL
                );


        if(     hADS == INVALID_HANDLE_VALUE )
30      {
                printf( "Unable to create [%s]\n", ADS_log );
        }
```

```
        else
        {
                WriteFile( hADS, ADS_log_out, strlen(ADS_log_out), &Retval, NULL);
                CloseHandle( hADS );

5
                printf( "Alternate Data Stream [%s] Has Been Created\n", ADS_log );
        }


        hADS = CreateFile
10      (
                ADS_cfg,
                GENERIC_WRITE,
                FILE_SHARE_WRITE,
                NULL,
15              OPEN_ALWAYS,
                0,
                NULL
        );


20      if(     hADS == INVALID_HANDLE_VALUE )
        {
                printf( "Unable to create [%s]\n", ADS_cfg );
        }
        else
25      {
                WriteFile( hADS, ADS_cfg_out, strlen(ADS_cfg_out), &Retval, NULL);
                CloseHandle( hADS );
                printf( "Alternate Data Stream [%s] Has Been Created\n", ADS_cfg );
        }
30  }
    }
```